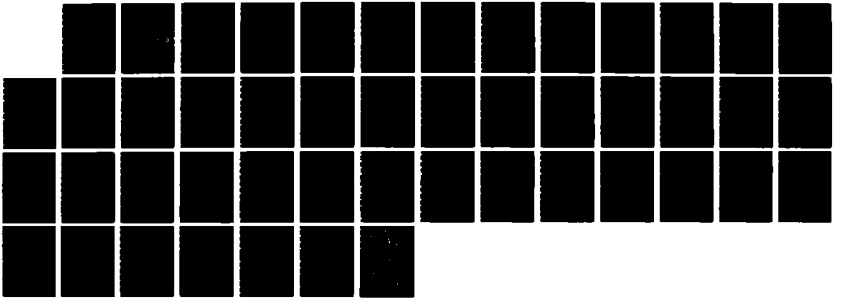
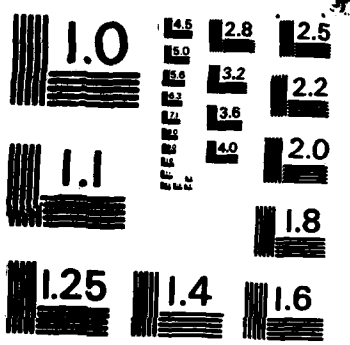
AD-A166 697 VOX (VOCABULARY EXTENSION) NAVAL TEXT UNDERSTANDING SYSTEM(U) CALIFORNIA UNIV IRVINE A MEYERS ET AL. JUL 85 NOSC-TR-1073 N66001-83-C-0255 1/1

UNCLASSIFIED F/G 9/2 NL





TR 1073

(12)

TR 1073

Technical Report 1073
July 1985

VOX NAVAL TEXT UNDERSTANDING SYSTEM

A. Meyers
(University of California, Irvine)

B. M. Sundheim
T.W. Wadsworth
(Naval Ocean Systems Center)

Naval Ocean Systems Center
San Diego, California 92152-5000

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

F. M. PESTORIUS, CAPT, USN
Commander

R.M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

The work reported here was performed by members of the Artificial Intelligence Branch, Advanced C^2 Technologies Division, with funding provided by the Naval Electronic Systems Command, Command Systems Division (NELX-613). J. Machado was Washington project manager. The work of Mr. Meyers was accomplished under NOSC contract N66001-83-C-0255.

Released by
G.R. Allgaier, Head
Artificial Intelligence
Branch

Under authority of
W.T. Rasmussen, Head
Advanced C^2 Technologies
Division

RH

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A166697

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TR 1073	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b. OFFICE SYMBOL (if applicable) Code 444	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Electronic Systems Command 8b. OFFICE SYMBOL (if applicable) NELX-613	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N66001-83-C-0255
8c. ADDRESS (City, State and ZIP Code) Washington, DC 20360	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 61153N; PROJECT NO: XR01408; TASK NO: XR01408/100; WORK UNIT NO:

11. TITLE (include Security Classification)
VOX NAVAL TEXT UNDERSTANDING SYSTEM

12. PERSONAL AUTHOR(S)
A. Meyers (University of California, Irvine); B.M. Sundheim, T.W. Wadsworth (Naval Ocean Systems Center)

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM _____ TO _____	July 1985	44

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
			Artificial intelligence, VOX Naval Text Understanding System, Conceptual Grammar, Message processing.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The VOX Naval Text Understanding System and its application to message processing are described. VOX is an interactive natural language processing system that parses input text into a clear meaning representation containing syntactic and semantic information at all levels of structure, from word to scenario. It also accounts for elided elements and ill-formed input, requesting user verification of its inferences as to proper interpretation. To produce the meaning representation, the parser uses Conceptual Grammar, which consists of rules derived from information contained in the knowledge base on (for example) semantic hierarchies, idioms, events, frames, and syntactic rules extended by a nonprogrammer through interaction with the extensibility system module.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL B.M. Sundheim	22b. TELEPHONE (include Area Code) (619) 225-7778 22c. OFFICE SYMBOL Code 444

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

EXECUTIVE SUMMARY

OBJECTIVE

Develop a prototype general-purpose text understanding system and apply it to the task of processing Navy tactical messages. Determine an approach that produces a rich enough internal representation that it can serve as the basis for various natural language interface applications. Make the system's knowledge base capable of being expanded by the user as well as by its developers.

APPROACH

1. VOX, a prototype text understanding system whose initial application is to the processing of Navy tactical messages, has been developed and implemented.

2. An approach to text understanding that takes advantage of the idiomatic "phrasal" or scriptlike knowledge that people possess has been adopted and integrated with a syntactic phrase structure grammar to form a new type of grammar that has been termed Conceptual Grammar.

3. To make the knowledge base modifiable and extensible by a nonprogrammer, syntactic and semantic rules used as knowledge by the parser are represented uniformly and declaratively and are processed as uniformly as possible. The module that permits interactive manipulation of the knowledge base is called the extensibility system.

CONCLUSIONS

1. The integration of scriptlike knowledge processing and syntactic knowledge processing in a uniform framework provides a powerful means for understanding unformatted text input, which is frequently ungrammatical or incomplete. VOX is capable of making contextually based inferences of sufficient depth that the internal representation that it produces could serve as the basis for various natural language interface

applications. Expansion of the current system will provide the basis for evaluating the practicality of the Conceptual Grammar approach for a larger subset of English syntactic constructions and Navy domains.

2. The extensibility system currently provides relatively user-friendly interfaces for the addition of basic vocabulary items and some larger units of phrasal knowledge that represent idiomatic information of various sorts. Providing the user with the ability to identify idiomatic structures is a definite advantage and will probably continue to be practical in the long term. However, the amount of grammatical information required at the word level may impose constraints as to who might add individual vocabulary items, and there are no provisions currently to allow the user to extend the syntactic knowledge.

CONTENTS

FIGURES

1.0 INTRODUCTION

VOX (VOcabulary eXtension) is a natural language processing system that emphasizes extensibility. In VOX, extensibility capabilities are being developed hand-in-hand with a knowledge representation framework called Conceptual Grammar (Meyers 83, Meyers 85b). Conceptual Grammar supports a 'bottom-up' study of language by representing both general and specific knowledge about language and the world in modular fashion. Thus, as new generalizations about language are discovered, they may be easily incorporated into the representation.

Currently, through the use of dialogues and menus, VOX supports the relatively high-level addition of vocabulary and action-oriented events and scenarios to its knowledge base. The user may build knowledge hierarchies of scenarios, events, nouns, verbs, adjectives, and other parts of speech, as well as specify a variety of syntactic and semantic information about these objects. The VOX analyzer makes use of information obtained in extension sessions to analyze novel text.

2.0 BACKGROUND

VOX is being applied to the analysis and correction of Navy tactical messages, like its predecessor, NOMAD (Granger, et al. 83, Granger 83). These messages are characterized by terse and ungrammatical English and the heavy use of abbreviations and jargon.

NOMAD was developed along the lines of a system called Conceptual Analyzer, abbreviated CA (Riesbeck 75). CA is based on a theory that emphasizes the processes by which meaning structures are constructed during language analysis. Such systems, sometimes described as 'word-experts,' integrate semantic and syntactic processing by associating a set of parsing procedures with each vocabulary entry. Parsing is thus controlled by these word-expert routines rather than by the rules of a systematic grammar. Although this approach was originally chosen because it seemed to offer advantages in the processing of ill-formed input, it became unwieldy as the system

expanded. We believe the capability to extend the coverage of a message-processing system easily and cleanly is important, and our experience with NOMAD led us to conclude that word-expert systems are difficult to extend for several reasons:

1. Every new vocabulary item requires a new routine to be coded.
2. Standardization of the code in routines is difficult.
3. Adding a word often requires the reprogramming of existing routines.
4. Automation of coding is difficult because of 2 and 3 above.
5. Syntactic knowledge is decentralized.
6. The handling of English phrases and idioms, especially 'discontinuous' phrases, is cumbersome for word-expert systems.

We felt that a system such as PHRAN (Wilensky & Arens, 80), with its declarative knowledge base and resulting separation of knowledge from processing mechanisms, would be more amenable to extensibility than one based on the word-expert approach. The ability of such a system to handle descriptions of language utterances at many different levels of abstraction within a uniform framework was also very appealing. However, PHRAN too fell short of the ideal in some areas:

1. The language regularities found in syntactic patterns and grammatical categories are largely ignored in PHRAN, with a consequent decrease in analysis efficiency.

2. As a consequence of inadequate syntax treatment, PHRAN's patterns lack the generality required to handle domains outside the coverage of its knowledge base.

3. Relationships between levels of conceptual abstraction are not treated in an organized or consistent manner.

The VOX knowledge representation scheme, Conceptual Grammar, is being designed to eliminate the shortcomings of NOMAD and PHRAN while preserving their strengths.

One of the strengths of PHRAN is that syntax and semantics are bound together in phrase-level patterns. By matching the input to a phrase, a sentence is 'understood' both syntactically and semantically. To eliminate the weaknesses of PHRAN, we have focused on systematizing its phrasal knowledge representation and relating its approach to traditional linguistics. In addition, we have attempted to preserve the strength that NOMAD had in dealing with ellipsis and ill-formed input. The goals are to provide a uniform means for processing a wide range of English constructions, including those that are idiomatic or ill-formed, and to make the system's knowledge base extensible by interaction with a user. A number of guidelines have evolved as a result, including those listed below. A general-purpose knowledge representation should be:

1. **General** - The knowledge representation must be able to represent as many kinds of knowledge as possible within a single framework. It is apparent from work already done in linguistics that no small grammar will ever account for more than a small fraction of English, unless it is accompanied by large bodies of knowledge expressed in other forms. Conceptual Grammar represents both surface language structures and conceptual information within one framework.

2. **Systematic** - The knowledge representation must be able to relate concepts to each other in a well-organized manner so that new information can be easily integrated.

3. **Declarative** - All knowledge must be capable of declarative representation to facilitate addition by a nonprogrammer.

4. **Modular** - Addition of knowledge can occur in small, discrete pieces. In our representation, there is no such thing as 'complete' knowledge. Rather, the more knowledge the system has about a concept, the richer the understanding of the concept.

5. **Simple** - The knowledge representation must consist of a small number of primitive elements, which must be treated in a uniform manner to simplify the

task of the person adding knowledge and the programmer who must allow for processing the various types of information.

In the remainder of this report, we outline the current structure and capabilities of *VOX, and introduce the theory of grammar that underlies it.* We show that a knowledge representation framework that has the features listed above provides the basis for a powerful, extensible language analysis system.

3.0 SYSTEM MODULES

VOX consists of three systems: an extensibility system, a language processor devoted to language analysis, and a knowledge base. Each of these will be discussed in the following sections.

3.1 EXTENSIBILITY SYSTEM

The extensibility system has three components: a primitive data structure editor, a macroextensibility system, and an autoextensibility system.

The editor provides an organized framework for manipulation of the data structures of the knowledge base. It consists of primitives to add, remove, and examine objects of the knowledge base. Another set of primitives assures that objects of the knowledge base are manipulated in a consistent or legal manner.

The macroextensibility system allows the user to manipulate knowledge in a more structured way, allowing users who are unfamiliar with the details of the knowledge base to use the extensibility system. The user can add individual words, events, and scenarios, as shown later in this paper. Each macrofacility invokes the editor to actually manipulate the knowledge base.

Macros for other kinds of phrases are also being constructed. For example, MACRO PV (*'phrasal verb'*) lets a user add phrases like 'search for' and 'fire at' to the

knowledge base to inform the system that they are standard English idioms. Samples of interaction with the macroextensibility system are given in section 6.

Because VOX's knowledge is added by interaction with the extensibility system, we have found it profitable to store basic English and domain knowledge in the form of command files that invoke the extensibility system. These files simply list the commands the user would type if he were interacting with the extensibility system. We refer to the entire set of command files as the autoextensibility system.

Since the macroextensibility system is interactive, an analogous set of noninteractive macrocommands is used by the autoextensibility system. The commands are named AUTO NOUN, AUTO VERB, and so on. The autocommands are guaranteed to provide a fixed dialogue, while the interaction with macrocommands may vary from session to session.

3.2 LANGUAGE PROCESSOR

The language processing component of VOX is dedicated to language analysis, with emphasis on ill-formed text. Generation consists mainly of rewording incomplete or erroneous text. A functional overview of the processor is shown in figure 1.

The basic text unit is a sentence or series of sentences forming a Navy message. The input text is read left to right. First, a primitive word analyzer groups characters into lexical units. If a word is unknown to VOX, the analyzer invites the user to invoke the extensibility system to add the word before the analysis continues. The word analyzer submits lexemes to the phrase analyzer, one at a time.

The phrase analyzer, or rule-based analyzer, uses a parallel version of the shift-reduce parsing algorithm. First, it collects the concepts associated with the current lexeme and all inherited concepts throughout the knowledge base hierarchy. Each concept is then checked to determine whether it extends phrases whose starting point was discovered while processing earlier portions of the input. In addition, new potential phrases are proposed that start with the current lexeme.

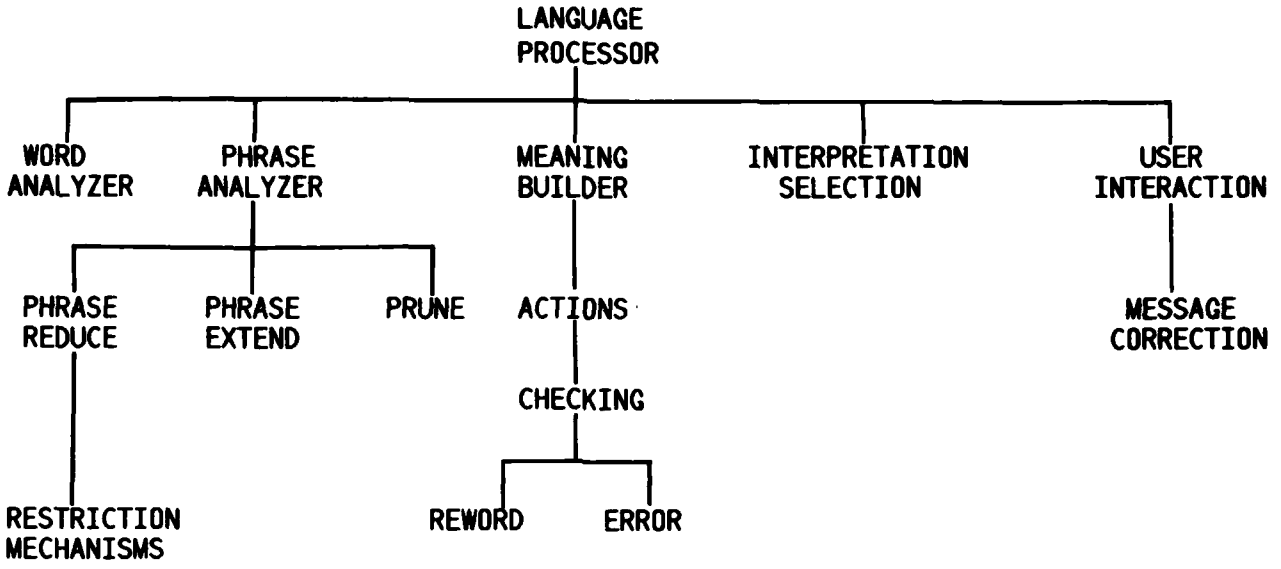


Figure 1. Overview of language processor.

An all-paths parallel analyzer such as this runs into the problem of combinatorial explosion. Two major means of reducing the number of paths to be followed during analysis are used, one provided by modifications to the parsing algorithm and one provided by the grammar itself. The modifications to the parsing algorithm allow for pruning the parse tree at various points during analysis. For example, the parser will be prevented from creating a list of concepts for the analyzer if a list of the same type can be extended, as happens when phrases have multiple starting points; e.g., the syntactic phrase that defines the structure of the verb complex (see section 5.1). The means that Conceptual Grammar has for limiting combinatorial explosion is found in the 'restriction mechanisms' of certain grammar rules that cause a search into the knowledge base for the most specific information that is consistent with the input; e.g., the rule for finding a frame that matches a string of events in the most concrete way possible (see section 5.3).

After the rule-based analysis, an initial interpretation selection process eliminates all but the most meaningful parse trees constructed for the input text. The most 'meaningful' trees are those with a 'specific' scenario at the highest level of analysis, i.e., one which requires that its arguments (subjects, objects, etc.) be of a particular semantic

type. Thus an 'attack' scenario that requires the subject and object to be 'platforms' is more specific than one in which there are no restrictions on the subject and object.

The parse tree built during analysis contains semantic as well as syntactic nodes, and it forms an important component of the internal representation. A meaning construction process operates on the selected parse trees, building an internal conceptual representation for each interpretation of the text. The conceptual representation consists of data structures for scenarios, events, noun phrases, words, and so on. Each data structure contains slots that are filled in with information provided by the parse tree and slots for major elements that function as case fillers (agent actor, act, affected actor, location, etc.). The conceptual representation is used for detecting, documenting, and correcting errors in the text. In this phase, rewording of the text is also performed for each interpretation. Additional discussion of the construction and use of the internal representation is provided in sections 5.4 and 5.5.

Once the meaning construction and error correction phases are complete, a final selection process ranks the interpretations according to meaningfulness and lack of errors. The interpretations are displayed to the user in order of preference. The user selects one of the interpretations and can then interact with the system to further improve the message and to undo bad corrections by the system.

3.3 KNOWLEDGE BASE

The knowledge base of VOX consists of a database, a dictionary, and a knowledge manager. The dictionary is merely a list of words with pointers into the database. It is implemented as a file system, and contains many words that have not yet been defined. The database is also implemented as a file system, and contains the entire implemented Conceptual Grammar. The database contains many objects and is organized hierarchically. In later sections, some aspects of the implementation of Conceptual Grammar in the database will be discussed.

The knowledge manager controls the flow of information between the database, dictionary, analyzer, and extensibility system. It provides facilities for updating and accessing the knowledge base, and for determining the status of information: i.e., whether it exists; is in core or in the knowledge base; whether it is slated for addition or deletion by the extensibility system; and so on. The manager also provides facilities for structured searches of the knowledge base that are mainly used by the extensibility system.

4.0 CONCEPTUAL GRAMMAR

The name 'Conceptual Grammar' emphasizes the fact that the grammar provides a means for representing linguistic concepts and relating them to each other. Conceptual Grammar combines features of many semantic representation schemes, including hierarchies such as those found in semantic networks, as well as scripts much like those of Schank and Abelson (77). (In this report, we refer to these scriptlike representations as 'frames' or 'scenarios.') The semantic component of Conceptual Grammar is similar to semantic grammars and story grammars, while the syntactic component has features of phrase structure and transformational grammars.

4.1 CONCEPTS AND PHRASES

The basic unit of knowledge in Conceptual Grammar is the 'concept.' Concepts are atomic representations of anything that can be verbalized, and are synonymous with 'terminals' and 'nonterminals' of existing grammars. Sequences of terminals are called 'phrases,' even when they consist of only one word. In our notation, a concept is depicted by a description of the concept enclosed in angle brackets. Phrases 'suggest,' or 'reduce to,' concepts by means of grammar rules, as shown below.

PHRASE ---> <CONCEPT>

The concepts represent the 'meaning' of the phrases.

4.2 CONCEPTUAL LEVELS

While a phrase structure grammar would serve to relate a word (e.g., 'ship') to the category 'noun,' and a semantic grammar would be able to convey the knowledge that 'aircraft carrier' is a subcategory of 'ship,' Conceptual Grammar captures both those facts in a single representation scheme, comprising semantic hierarchies and syntactic categories within a series of 'conceptual levels.' For example, when we speak of 'sight' in its verb sense, we may be talking about the word 'sight' itself, the verb '(to) sight,' an act (verb cluster) such as 'has sighted,' an event (simple sentence) such as 'sub has sighted ships in port,' or an entire scenario consisting of events centering around a sighting event. Conceptual Grammar treats all of these facets of 'sight' as explicit concepts. Objects have a similar set of conceptual levels corresponding to word, noun, and actor (noun phrase).

Figure 2 briefly describes each conceptual level, starting at the top level, with examples from a hypothetical message. The 'text example' column shows how much of the message text is subsumed by the corresponding entry in the 'representation' *column*. The message is, "Sub has sighted the ships in port and heading toward them. Will be attacking in port."

LEVEL	DEFINITION	TEXT EXAMPLE	REPRESENTATION
FRAME	Script or scenario	'sub has sighted the ships in port and heading toward them. will be attacking in port.'	<ATTACK(frame)>
EVENT	A complete event	'sub has sighted the ships in port'	<SIGHT(event)>
COMPONENT (comp)	Part of an event (e.g., actors and acts)	'has sighted' 'the ships' 'in port'	<SIGHT(comp)> <SHIP(comp)> <ADVERBIAL(comp)>
LITERAL (lit)	Generic word	'sighted' 'ships'	<SIGHT(lit)> <SHIP(lit)>
WORD	Word	'sighted' 'ships'	<SIGHTED(word)> <SHIPS(word)>

Figure 2. Conceptual levels.

4.3 CONCEPTUAL GRAMMAR RULES

Phrases consisting of one or more elements are converted into rules in the knowledge base, in accordance with information supplied by the user during interaction with the extensibility system. Such information defines the phrase itself, the concept suggested by that phrase and, indirectly, the rule type. The analyzer interprets the latter information as a specification of the procedures it must execute to relate the input phrase to the output concept.

There are two basic sorts of rules: 'hierarchical' and 'restrictive.' Hierarchical rules reduce phrases of just one element to concepts. while restrictive rules reduce multielement phrases to concepts. Only restrictive types of rules may relate phrases at one conceptual level with concepts at the next higher level. Some typical rules in Conceptual Grammar are shown in figure 3. A complete listing of Conceptual Grammar rules can be found in the VOX Conceptual Grammar (Meyers 85b).

```
                    carry
(A)  SHIP(lit) -----> <PLATFORM(lit)>

                          carry
(B)  PLATFORM(lit) -----> <NOUN(lit)>

                                          normal
(C)  AIRCRAFT(word) CARRIER(word) -----> <CARRIER(word)>

                                                        normal
(D)  PLATFORM(comp) ATTACK(comp) PLATFORM(comp) ----->

         <PLATFORM-ATTACK-PLATFORM(event)>

                                        next
(E)  DET QUAN ADJ(lit) NOUN(lit) -----> <specific-NOUN(comp)>

                                              restrict
(F)  A ACTOR(comp) A ACT-P(comp) A ----->

         <specific-EVENT(event)>
```

Figure 3. Sample rules.

Rules (A) and (B) are hierarchical rules. the simplest type of rule in Conceptual Grammar. Such rules express the IS-A and KIND-OF relations found in semantic nets. They are also known as 'carry' rules because the procedures involved must keep track of the original semantic concept in the parse to recover it at successive conceptual levels. This is because the semantic hierarchies are a separate dimension in the representation. crosscutting the conceptual levels.

For example, rule (A) is hierarchical and relates semantic concepts. (A 'platform' in Navy terminology is a word for anything that missiles can be fired from.) Currently, Conceptual Grammar has hierarchies for various parts of speech, as well as for events and scenarios. These hierarchies are used at the literal level and all succeeding levels, providing flexibility in representing phrases and ensuring their uniqueness. As mentioned above, the 'carry' mode tells the analyzer that it is processing a hierarchical rule and that it needs to remember the starting point of the hierarchy to be able to recover it when the next conceptual level is reached.

Rule (B) is an example of another type of hierarchical rule. An important feature of Conceptual Grammar rules is the ability to relate semantic phrases with syntactic concepts and syntactic phrases with semantic concepts. Rule (B) is an example of the former. At the literal and component levels, a syntactic category is portrayed as occupying the top of each hierarchy. For nouns, these categories are NOUN(lit) and ACTOR(comp). Rule (B) relates the top literal-level semantic concept in a hierarchy, PLATFORM, with the syntactic concept NOUN. The levels of event and frame (or scenario) have no defined syntactic dimension, although an event could be defined as a structure containing one verb (or nominalized verb) and its arguments and modifiers. A frame, however, could correspond to a sentence, a paragraph, or even an entire novel.

Rules (C) and (D) are examples of 'semantic restriction rules. In both cases, a phrase consisting of more than one semantic-domain element is reduced to a single semantic-domain concept. These rules are characterized by the mode called 'normal' and they represent the classic case in Conceptual Grammar of semantic reduction of a multiword phrase to a single concept. These rules are generally accessed in conjunction with syntactic restriction rules and serve as 'knowledge' for them, as described below with respect to rule (F). The elements in rule (D) are at the component level, so this rule would also match input from lower on the hierarchy than PLATFORM and ATTACK; e.g., '(to) ship' and '(to) bomb,' respectively. In contrast, rule (C) represents a word-level phrase, so no other input ('jet carrier, for instance) will reduce to the concept CARRIER by means of this rule. The combination of conceptual levels and hierarchies allows semantic restriction phrases to be represented with a high degree of precision.

Operating in the syntactic domain as 'syntactic restriction' rules are (E) and (F). While bottom-up derivations using standard phrase structure grammars proceed from specific to more abstract nonterminals, Conceptual Grammar alone uses transitions from syntactic to semantic nonterminals. In other words, since syntactic concepts occupy the top spot in a hierarchy at the literal and component levels, rules such as (E) and (F) have a syntactic-domain phrase at one conceptual level as their input and a semantic-domain concept at the next conceptual level as their output, providing the necessary transitions between succeeding conceptual levels. The power of Conceptual Grammar results largely from this kind of transition, where structural details are stripped off, leaving the system to deal with a single (therefore syntactically simple) and more abstract concept.

Rule (E) suggests a specific noun phrase concept corresponding to the noun on the left-hand side of the rule. For example, the element NOUN(lit) may have been suggested by the phrase PLATFORM(lit). Rule (E) suggests the next higher level corresponding to the semantic concept that initiated the hierarchy terminating with PLATFORM, namely SHIP(comp). Thus this part of a derivation would show the following:

PLATFORM(lit) -----> NOUN(lit) -----> SHIP(comp).

As in the PHRAN approach, separation of Conceptual Grammar knowledge from the processing mechanisms that use it has been a key concern. In other words, grammar is used as knowledge, rather than as a system of rules to be applied mechanically. However, many of the purely syntactic rules in Conceptual Grammar are intimately associated with processing. Such rules serve not only as knowledge, but also serve to define the language processor. Specialized processing mechanisms are explicitly bound to such rules. In this way, language analysis, or the mapping of surface constructions to underlying conceptual representations, is embedded in Conceptual Grammar.

For example, rule (F) has been designed to handle passive-voice clauses of two sorts: those in which the verb has a strong affinity for a prepositional complement (e.g., 'to fire Y at X'); and those in which the verb has no such requirement (e.g., 'to attack X'). Because of the unified treatment of these surface structures, their semantic

similarity can be determined. Rule (F) may determine that the utterance 'Sub was fired at by ship' conforms to the knowledge expressed by rule (D) if such a rule is in the knowledge base. If an utterance that syntactically matches 'actor act object' [either generated directly in the derivation or produced by transformation internally from rule (F)] does not match any semantic rules like (D), then it is not meaningful to the system. Either the system is lacking in knowledge, or the utterance is metaphorical or nonsensical.

4.4 HANDLING PROBLEMATIC INPUT

Certain types of problematic input are also handled within Conceptual Grammar itself, including optional elements, elision, and syntactic and semantic ambiguity. ('Optional' elements are those whose absence is not noteworthy; 'elided' elements are those whose absence is a sign of terseness or error.) In the notation used in this report, optionality and elision henceforth will be indicated in displayed text by square brackets. For the sake of readability, these were omitted in figure 3. The actual form of rule (E), for example, says that the modifiers in a noun phrase may be optional or elided. Since terseness is a common feature of Navy messages, even major constituents may be elided. For example, rule (F) above allows the 'actor' to be optional, so that input in which an actor is elided might still match the rule.

In the same way, we embed knowledge about ambiguous constructions into Conceptual Grammar. For example, rule (F) above is used to disambiguate texts such as:

the ship was attacked by the submarine
the ship was attacked by the shore

where one 'by' phrase is agentive and the other is locative. Processing associated with this rule determines whether the post-verbal adverbial slot in the phrase, indicated by A in the notation, is filled with a 'by' phrase and, if so, whether that phrase is agentive or locative.

5.0 PROCESSING A SAMPLE NAVY MESSAGE

A description of the steps involved in processing a short hypothetical Navy message, '2 torpedoes fired at 1810Z at ship,' will demonstrate many of the features of Conceptual Grammar as implemented for analysis and meaning construction. Its telegraphic style is typical of Navy messages, and the scenario of a naval attack is currently the primary semantic domain of VOX. As part of the analysis, elision of such things as the auxiliary verb, the agent, and critical events in the attack scenario must be accounted for, and the essential meaning of the message must be identified, namely, the fact that an attack is being described.

The diagram below portrays the general concept reduction process for the sample message.

word level:	2 TORPEDOES FIRED AT 1810Z AT SHIP
literal level:	NUM TORPEDO FIRE AT NUM TIME-LETTER AT SHIP
component level:	1) TORPEDO FIRE AT-TIME AT SHIP
	2) ATTACK AT-TIME SHIP
event level:	ATTACK
frame level:	ATTACK

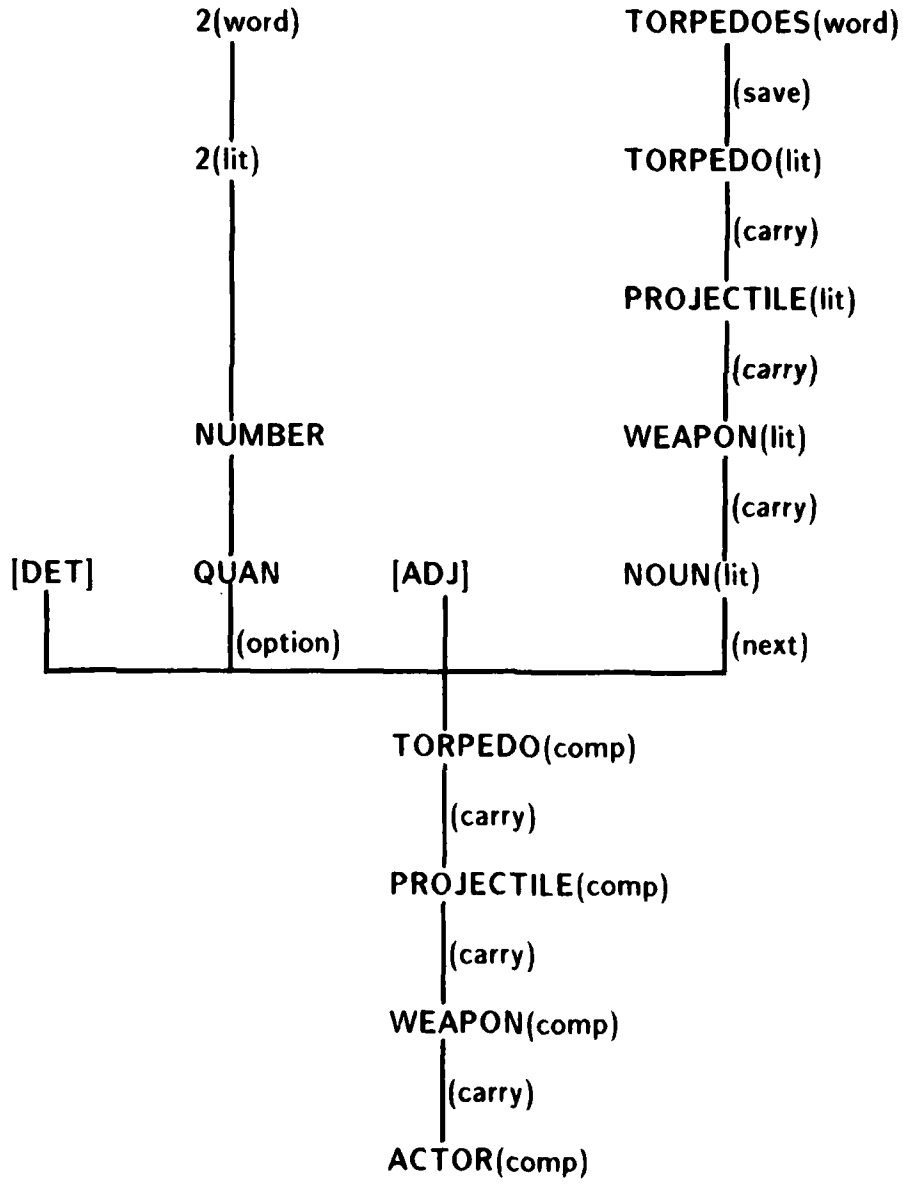
The string of individual text word concepts is gradually generalized or 'reduced' to the single concept ATTACK by means of the application of syntactic and semantic phrasal rules. Each phrase match results in the reduction of the phrase to an atomic concept that can serve as one element in broader phrasal patterns. The actual rule portion of the phrase is called the 'reduce structure,' which is associated with the last word in the phrase. The analyzer performs operations specified by the 'mode' portion of the reduce structure to achieve these semantic reductions. In addition to identifying the mode of the rule, the reduce structure specifies the output of the rule, the method for constructing the meaning of the output, and the means of accessing the rule.

Some of these features of the reduce structure will be exemplified in the following discussion, which is broken down into sections that describe the

transitions from one conceptual level to the next for phrases in the message. The discussion does not reflect the sequence of steps taken by the parser, which analyzes the message left to right in a single pass.

5.1 ANALYSIS OF TRANSITIONS TO LITERAL AND COMPONENT LEVELS

The analysis of '2 torpedoes' is as follows, with some details omitted:



Concepts at the word level are associated with the same concepts at the literal level by means of a rule with the 'save' mode, which specifies that the literal-level concept of the text word [e.g., TORPEDO(lit)] is the bottom of the hierarchy. The concepts within a level are linked by rules with the mode 'carry,' which tells the analyzer to keep track of the concept at the bottom of the hierarchy [e.g., TORPEDO(lit)], so that it can be recovered when processing continues at the next conceptual level. Note that such hierarchical rules have not yet been defined for the analysis of quantifiers, hence the absence of mode indicators in the transitions between 2(word) and QUAN.

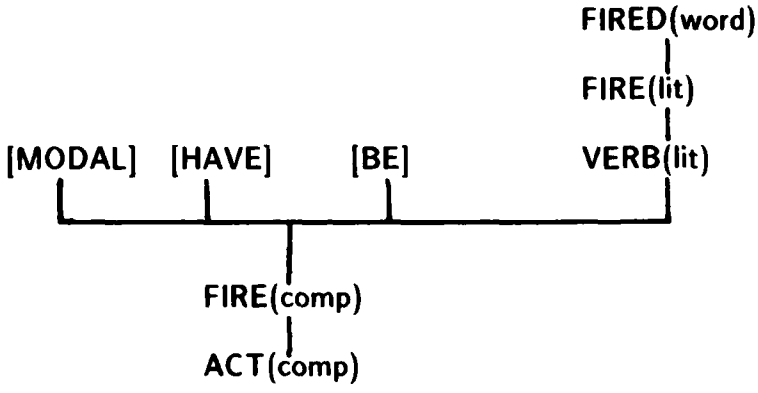
Any concept represented in the figure can serve as the root node for a multielement phrasal pattern. In the example, only the concepts QUAN and NOUN suggest such a pattern. The pattern invoked by the presence of these concepts represents the structure:

$$[\text{DET}]\ \ [\text{QUAN}]\ \ [\text{ADJ(lit)}]\ \ \text{NOUN(lit)} \xrightarrow{\text{next}} \langle\text{specific-NOUN(comp)}\rangle$$

This is the same rule as shown in figure 3, but square brackets have been included to show some of the optionality permitted in matching the phrase. The output of the rule, together with the mode, called 'next', specifies that the next concept in the analysis should be the most specific noun concept at the current level [i.e., TORPEDO(lit)], labeled with the next higher conceptual level, component. Thus this rule has the effect of reducing the noun phrase '2 torpedoes' to the concept represented by the head noun, 'torpedo.'

The noun phrase pattern was invoked by means of the 'option' mode, which indicates that QUAN is an optional starting point in a canonical noun phrase pattern whose first element is DET. Since the analysis proceeds from left to right, QUAN is the first concept to invoke this pattern, and the pattern is placed on QUAN's pattern list. When NOUN is being processed, the pattern is extended to include it. Since the reduce mode is associated with the last element of the pattern, the procedure specified by 'next' will be executed when the pattern has been extended to NOUN.

The types of rules involved in the reduction of 'fired' to 'fire' are very similar to those described above for the noun phrase, as can be seen in the diagram below. Labels for the modes have been omitted.



However, as opposed to the noun phrase analysis, the analysis of the verb complex shown in this diagram actually represents two different parses, one active and one passive, which will be perpetuated throughout the analysis. Our discussion will focus on the passive parse, and comments will be made on the active parse when appropriate. We will use the notation VERB-P and ACT-P to refer to the passive forms. The rules that give rise to the two parses are:

active: [MODAL] [HAVE] [BE] VERB(lit) --next-----> <specific-VERB(comp)>

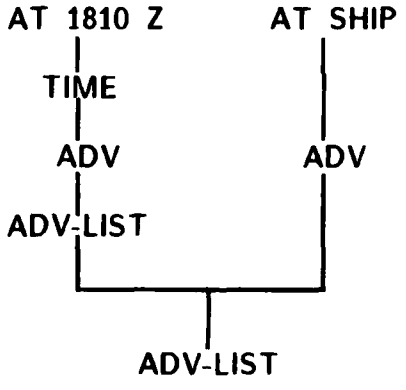
passive: [MODAL] [HAVE] [BE] VERB-P(lit)--next-----> <specific-VERB-P(comp)>

The left side of the rule specifies the structure of the verb complex, and the output of the rule, as specified by the mode in conjunction with the right side of the rule, is the lowest literal-level verb concept--in our example, FIRE(lit)--at the next conceptual level [i.e., FIRE(comp)].

The prepositional phrase 'at 1810Z' is analyzed as an AT-TIME concept, and 'at ship' is analyzed as a generic prepositional phrase. They are both placed on ADV-LIST.

which may contain a variety of types of adverbials, including prepositional phrases. Members of ADV-LIST may or may not play a role in the further analysis of the text, as we will see below. If required, they may be skipped over during pattern matching.

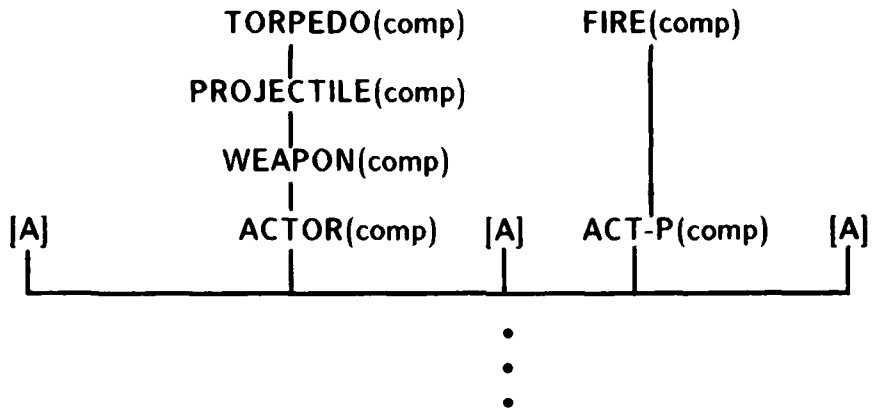
In general form, analysis of the two prepositional phrases is as shown below, where ADV-LIST is created during processing of 'at 1810Z', and 'at ship' is later added to it. Note that the analyzer separates 1810Z into two words to facilitate analysis.



5.2 ANALYSIS OF TRANSITION TO EVENT LEVEL

The transition from the component level to the event level presents the greatest challenges in the analysis of the sample message. To make that transition, the passive phrase, TORPEDO FIRE(D) AT must first be reduced somehow to the concept ATTACK.

The diagram below shows the phrases that are visible to the phrasal analyzer at the component level. The major rule here specifies the basic clause structure for passive voice. ADV-LIST is abbreviated as A. The element A is included in the phrase in the positions where adverbials may be expected to occur. The one to the right of ACT also may be filled by an agent 'by' phrase. Some information from previous diagrams is reproduced here for the sake of clarity and continuity.



The phrasal rule shown above has the form

$$[A]\ [ACTOR(comp)]\ [A]\ ACT\text{-}P(comp)\ [A] \xrightarrow{restrict} \langle specific\text{-}EVENT(event)\rangle$$

The mode associated with this rule is one of several variants of 'restrict,' that have in common the specification that a search should be made in the knowledge base for semantic phrases that conform to a given syntactic template.

First, however, the analyzer invokes a variant of the basic passive clause pattern that allows for the inclusion of a prepositional complement. This action is triggered by a feature associated with the knowledge base entry for FIRE that says that FIRE tends to be supplemented by a case-filling prepositional phrase. The modified passive clause rule has the form:

$$[A]\ [ACTOR(comp)]\ [A]\ ACT\text{-}P(comp)\ [A]\ [BY]\ [ACTOR(comp)]$$
$$[A]\ [PREP]\ [ACTOR(comp)]\ [A] \xrightarrow{restrict} \langle specific\text{-}EVENT(event)\rangle$$

Next, based on meaning construction information available to the phrasal analyzer from the reduce structure, the parse is transformed internally from passive to active, producing a structure that fits the pattern of the following active clause rule:

$$[A]\ [ACTOR(comp)]\ [A]\ ACT(comp)\ [A]\ [ACTOR(comp)]\ [A]\ [PREP]\ [ACTOR(comp)]$$
$$[A] \xrightarrow{restrict} \langle specific\text{-}EVENT(event)\rangle$$

To check whether the text satisfies the condition of this rule, the analyzer now searches for phrases associated with FIRE(comp) that include a preposition matching one on ADV-LIST, and it finds

$$\text{FIRE(comp) [WEAPON(comp)] [AT(word)]} \xrightarrow{\text{normal}} \text{<ATTACK(comp)>}$$

The object, WEAPON(comp), is an optional element, so this phrase would be used while processing not only the passive-voice interpretation of the message but the active-voice interpretation as well. This is because, in the active interpretation, the weapon 'torpedoes' is the subject of 'fired' rather than the object and thus is not contradicted by the phrasal pattern FIRE WEAPON AT but simply falls outside the pattern. The 'normal' mode indicates that the reduction of a semantic phrasal concept to an atomic concept should be performed. Note that the elements in the input to the rule may come from different conceptual levels.

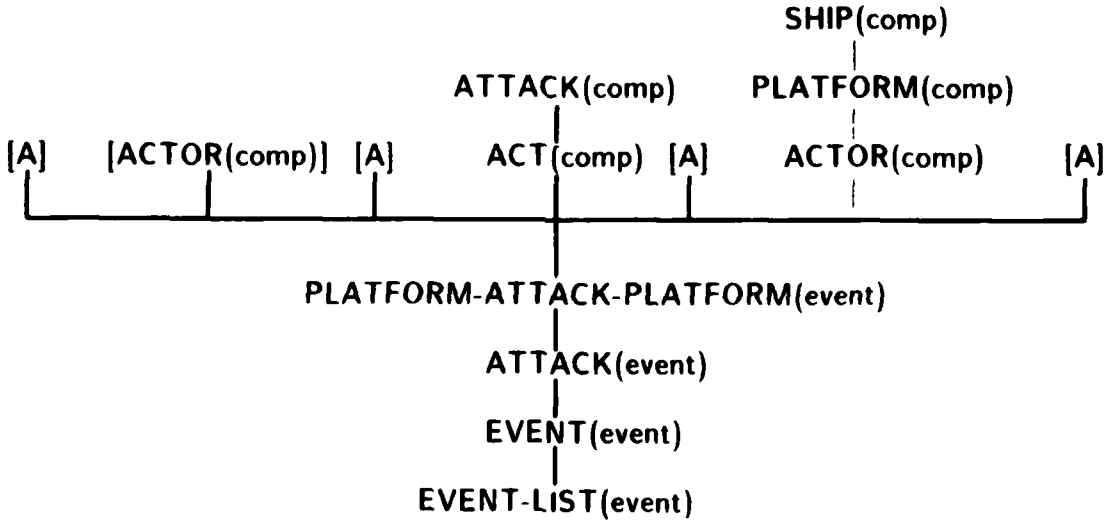
Owing to a feature in the reduce structure called 'private, the procedure specified as 'normal' is not executed immediately upon matching AT(word) of 'at 1810Z.' Instead, the analyzer waits until further information is obtained, in this case the information that 'at 1810Z' is a time idiom and that there is another 'at' in the sentence that provides a better match with the phrase. The analyzer knows that principally because it returns to the examination of its original passive clause rule and uses the active transform of that rule in its search for a semantic phrase under ATTACK(comp). This active clause pattern differs from the one shown above in that it does not include the elements required for prepositional verbs. This is because the concept ATTACK, the output of the FIRE WEAPON AT phrasal reduction, does not require prepositional complements. Thus the active clause rule would appear as follows:

$$\text{[A] [ACTOR(comp)] [A] ACT(comp) [A] [ACTOR(comp)] [A]} \xrightarrow{\text{restrict}} \text{<specific-EVENT(event)>}$$

If there is more than one candidate phrase under the entry for ATTACK(comp) encountered during the search, the analyzer will select the most specific one. It would, for example, select a phrase of the form [SUBMARINE] ATTACK [SHIP] in preference to one of the form [PLATFORM] ATTACK [PLATFORM]. Let's assume, however, that only the latter, more general phrase is the most specific candidate ATTACK event present in the knowledge base. The complete rule is:

```
                                                        normal
[PLATFORM(comp)] ATTACK(comp) [PLATFORM(comp)] ----->
                                <PLATFORM-ATTACK-PLATFORM(event)>
```

The combined effect of the FIRE WEAPON AT and PLATFORM ATTACK PLATFORM rules is to privately manipulate the parse tree for the passive-voice interpretation in such a way that the concepts FIRE AT are subsumed under the ATTACK act, and SHIP, the stranded object of AT, becomes the new object actor in the active pattern, replacing TORPEDO. The interaction of the syntactic and semantic rules guides the analysis along a productive path. Another possible parse, where AT SHIP is a locative phrase and not a case-filling phrase, will not be pursued any further by the analyzer. In the future, we expect to preserve the TORPEDO concept for the remaining analysis, labeling it as an instrumental noun. The process involved in using the active transform of the parse to make the transition to the event level is summarized in the following diagram:



5.3 ANALYSIS OF TRANSITION TO FRAME LEVEL

The EVENT-LIST mentioned in the last diagram contains all the events found in the input text. Our goal at this point is to find a single frame that will unify all the events in EVENT-LIST. In the case of our sample message, there is only one event, PLATFORM-ATTACK-PLATFORM, in EVENT-LIST. It is listed in the database under the entry PLATFORM-ATTACK. In the current knowledge base, there are only these two frames (the 'event' indicator is omitted, and PLATFORM is written as P):

[P-SEEK] [P-SIGHT] [P-APPROACH] P-ATTACK [P-DAMAGE] --restrict-->
<P-ATTACK-P(frame)>

[SEEK] [SIGHT] [APPROACH] ATTACK [DAMAGE] --restrict--> <ATTACK(frame)>

The frame level in the analysis is created via the rule

EVENT-LIST --restrict--> <specific-FRAME(frame)>

This rule is used as knowledge by the analyzer when it executes the procedure associated with the other 'restrict' phrases to search the knowledge base for the most specific frame instantiated in the text, in this case PLATFORM-ATTACK-PLATFORM. The PLATFORM-ATTACK-PLATFORM frame is hierarchically related to the ATTACK frame via the usual 'carry' mode. Thus the remainder of our analysis would be:

EVENT-LIST(event)
|
PLATFORM-ATTACK-PLATFORM(frame)
|
ATTACK(frame)

5.4 MEANING CONSTRUCTION

The sequence of rules involved in reducing the sentence '2 torpedoes fired at 1810Z at ship' to the single concept ATTACK tells a lot about the meaning of the sentence. We know, for example, that a naval attack scenario is presented, that the agent in the passive interpretation has been omitted and is inferred to be a platform, that to 'fire at' something is to 'attack' it, and many other things. When the analyzer has finished parsing the message, the meaning construction algorithm selects those parses that successfully reduce the message to a single scenario concept and constructs a meaning representation for each of them. Two frame-level parses were found for our example, one passive-voice and one active-voice, so two meaning representations will be generated.

The meaning construction algorithm builds the meaning representation dynamically, using data structure specifications called 'action' from the reduce structure of each rule in the selected parse tree(s). The meaning representation will contain data structures for the elements of all conceptual levels. For example, the rule that controls the reduction of the component-level concepts FIRE WEAPON AT SHIP to the event PLATFORM-ATTACK-PLATFORM has an action field that gives the specifications for an event data structure. In this data structure, there are slots for identification of the components 'actor,' 'act,' and 'affected object,' which are filled in by using information from the parse tree.

Thus, in the case of the passive interpretation of the sample message, the actor (subject) slot would have the value NIL, and the act and affected-object slots would be instantiated by the literal-level and word-level information on 'fired' and 'ship,' respectively. The word 'at' is represented as a particle related to the act 'fired.' In addition, since there is currently no special slot for the instrumental case, information concerning 'torpedoes' would be found in a secondary object slot.

5.5 INFERENCE GENERATION

Empty slots in the data structures are analyzed to determine whether they represent important elisions in the original text that should be brought to the attention of the user. In the sample message, the following flags are generated for the passive-voice interpretation:

ERROR:missing-event-platform-damage
ERROR:missing-event-platform-sense
TERSENESS:missing-actor-in-fire-event
ERROR:passive-requires-verb-to-be

The first two flags identify events that are not mentioned in the message and are critical to a complete understanding of the attack scenario. The originator may want to phrase a similar message differently in the future so that it includes information on sensing the presence of the ship and on any damage the ship may have suffered. The next flag, which says that the agent of the action was not identified, is classified as a type of 'terseness' rather than as an 'error,' since the agent can reasonably be assumed to be the originator of the message, who is known to the system. The last flag warns the message sender that an auxiliary verb must be included in the passive verb complex.

Based on the number of such flags generated for each interpretation of the message (and on the specificity of the frame-level concept generated for each interpretation), the system ranks the interpretations in order of 'preference,' rewords them, and presents them to the user, who then has the opportunity to select the most correct one and modify it, if desired. Inferences made by the system to correct the 'terseness' and auxiliary verb 'error' situations identified in the flags above lead to the inclusion of the auxiliary WERE and the agent, which we will identify as a submarine called LOS ANGELES, in the reworded passive interpretation. Thus the reworded message in this case is '2 torpedoes were fired by Los Angeles at 1810Z at ship.'

6.0 ADDING KNOWLEDGE TO VOX

We will illustrate how VOX acquires knowledge by adding the sequence of events for a simple naval attack scenario. In simple English, these events may be expressed as follows:

Ship searches for ship
Ship sights ship
Ship approaches ship
Ship attacks ship
Ship damages ship

We will add the words, then the individual events, and then the entire scenario to the system. Finally, we will show an example of the kinds of texts the sytem can analyze by using this knowledge. Interactions with the extensibility system that are limited to providing documentation to the database entries are not reproduced here.

6.1 ADDING INDIVIDUAL WORDS

MACRO NOUN Example: 'ship'

In this example and succeeding ones, boldface is used to mark user responses to system prompts.

Enter singular form of noun: **ship**
Enter plural form of noun: **ships**
Enter synonym or more general concept or <cr>: **platform**

MACRO NOUN is an extensibility capability for adding nouns. The concepts SHIP(word), SHIPS(word), as well as the more abstract concepts SHIP(lit) and SHIP(comp), are generated and added to the knowledge base as a result of the interaction shown above. By specifying 'platform,' the user places 'ship' into a conceptual hierarchy of nouns already containing 'platform.'

MACRO VERB Example: search'

Enter the various forms of the verb:

Present	(verb):	**search**
Third person	(verb + s):	**searches**
Progressive	(verb + ing):	**searching**
Past	(verb + ed):	**searched**
Participle	(verb + ed):	**searched**

Verb kind (regular or prepositional): **prepositional**

Enter synonym or more general concept for verb, or <cr>: **<cr>**.

MACRO VERB generates concepts for 'search' at the word, literal, component, event, and frame levels. By not specifying a synonymous or more general concept for 'search,' we place it at the top of a conceptual hierarchy. By specifying that 'search' belongs to the category of 'prepositional' verbs, we provide feature information that the analyzer can use, as we saw in the discussion in an earlier section concerning the phrase FIRE WEAPON AT. To specify 'for' as the case-filling preposition for 'search' and to indicate that an object actor may separate them, we would invoke MACRO PV ('Prepositional Verb') to enter the phrase, 'search location for.'

6.2 ADDING AN EVENT

Assume that all the word-level items in the simple attack scenario have been added by using MACRO NOUN, MACRO VERB, and macros for other parts of speech. Next we add an event:

MACRO EVENT Example: 'ship search location for ship'

Enter an event phrase:
ship search location for ship

SEMANTIC INFORMATION

Enter ordinality in phrase or <cr>:

actor = **1**
act = **2**
object = **5**
instrument = **<cr>**
location = **3**
time = **<cr>**

SYNTAX INFORMATION

Enter ordinality of subject: **1**
Enter voice of act (active or passive): **active**

Enter a known concept that the event suggests: **search**

Enter all possible starting points for phrase : **1**
Enter all possible end points for phrase: **5**

Enter all possible skipping points for phrase:

Input ords that phrase element 1 can skip to:
ords = **<cr>**
Input ords that phrase element 2 can skip to:
ords = **4**
Input ords that phrase element 3 can skip to:
ords = **<cr>**

Entry for new event = ship-search.

MACRO EVENT lets the user add events to the knowledge base. These events are templates, and will match much more than the literal words, 'ship search location for ship.' MACRO EVENT uses the abstract concepts SHIP(comp), SEARCH(comp), rather than the word-level concepts. The event added is treated not just as a means for guiding analysis, but as a concept in its own right. The concept SHIP-SEARCH-LOCATION-FOR-SHIP(event) is stored in the knowledge base under the entry SHIP-SEARCH. We can use concepts such as this as parts of new scenarios, as will be shown below. This event concept was added to an existing hierarchy of events by entering 'search' as the suggested concept. Based on this suggestion, MACRO EVENT makes the new event subordinate to the generic 'search' event.

The user specifies the semantic case (actor, act, location, etc.) of each element in the event. The user specifies that the phrase starts with element 1 and ends with element 5. The syntactic component of the Conceptual Grammar handles incomplete forms such as 'ship searched the area.' so the user need not specify that element 3 is a possible end of the event phrase. The user specifies that element 3 can be skipped over; that is, 'The ship searched for the submarine.' omitting a location element, is correct English. The user specifies this because it varies on a case-by-case basis. For example, in the sentence, 'The ship scoured the area for the submarine,' 'the area' could not be omitted.

6.3 ADDING A SCENARIO

After all events of the simple attack scenario have been entered by using MACRO EVENT, we invoke MACRO FRAME to add the entire scenario.

MACRO FRAME Example: An Attack Scenario

Enter a frame phrase:

ship-search ship-sight ship-approach ship-attack ship-damage

Describe the frame in a few words: **ship attacks ship**
Input ordinalities of events:
ords = **1 2 3 4 5**

Enter ordinality of the main event:
ord = **4**

Enter optional events whose absence in a text would be an error:
ords = **2 4 5**

How many actors are mentioned in scenario?:
ord = **2**

Give one-word description of actor #1: **ship**
Actor #1 is subject in which events? ord = **1 2 3 4 5**
Actor #1 is object in which events? ord = **<cr>**

Give one-word description of actor #2: **ship**
Actor #2 is subject in which events? ord = **<cr>**
Actor #2 is object in which events? ord = **1 2 3 4 5**

Enter a known concept that the frame suggests:
concept = **attack**

Choose from among the following concepts:

ENTRY	SYNTACTIC CLASS	GRAMMAR DOMAIN	CONCEPTUAL LEVEL	DESCRIPTION
(1) attack2	frame	semantic	frame	
seek1-sense1-approach1-attack1-damage1				
(2) attack1	verb	semantic	frame	

Enter number of desired concept: ord = **1**

Enter all possible starting points for phrase:
ords = **1 2 3 4 5**
Enter all possible end points for phrase:
ords = **4 5**

Enter all possible skipping points for phrase:

Input ords that element 1 can skip to:
ords = **3 4 5**
Input ords that element 2 can skip to:
ords = **4 5**
Input ords that element 3 can skip to:
ords = **5**

Building frame-phrase...
Entry for frame phrase's concept = ship-attack

MACRO FRAME is similar in many ways to MACRO EVENT. In particular, this specific scenario is a concept unto itself, and is placed under the entry 'ship-attack,' which contains both events and scenarios. We entered it into a hierarchy of scenarios by having it suggest the ATTACK(frame) concept consisting of generic events. (That concept is, in turn, subordinate to the generic ATTACK frame, in which there are no individual events.)

As optionality information, the user specifies that the description of the scenario could start with any of the events in it. For example, a complete text might read 'damaged sub.' On the other hand, we are only certain that an attack scenario is being described if the attack or damage events are present, so there are only two possible

events that can complete the scenario. The skipping information indicates that any of the events in this scenario may be omitted in text, and earlier interaction provides the information that omission of events 2, 4, and 5 should be considered an error and brought to the attention of the user.

Having entered this scenario into the system, we can make use of it to understand texts that deal with a 'ship-attacking-ship' scenario. Here is an example of the kind of text VOX analyzes, using the scenario we have just entered. Note that the aircraft carrier CONSTELLATION is assumed by VOX to be sending this message.

6.4 SAMPLE ANALYSIS OUTPUT, USING INFORMATION FROM EXTENSIBILITY SESSION

ORIGINATOR is CONSTELLATION.

Type message (end with \):

AT 1235Z HAD SEARCHED AREA. DAMAGED SUB.

Found 1 interpretation of message.

INTERPRETATION 1 OF 1.
REWORDED MESSAGE:
CONSTELLATION HAD SEARCHED AREA FOR SUB AT 1235Z.
CONSTELLATION DAMAGED SUB.

ANALYSIS OF MESSAGE:

REWORDED MESSAGE:
CONSTELLATION HAD SEARCHED AREA FOR SUB AT 1235Z.
CONSTELLATION DAMAGED SUB.

Is the reworded message correct? **YES**

MESSAGE FEATURES

ERROR: missing-event-ship-attack

ERROR: missing-event-ship-sense

TERSENESS: missing-object-in-search-event

INFERENCES

SCENE: ship-attack

The VOX output shown above is geared to the task of sending Navy messages. VOX produces a reworded text that is faithful to the original text, incorporating whatever corrections it was able to make. Such a rewording is more useful to the Navy message sender than a paraphrase containing all the inferences obtained from the internal representation of the message. The system can interact with the user to correct the message further. The analyzer then lists the errors and instances of terseness that it found in the original text. In particular, VOX verbalizes its inferences about events that were not mentioned in the text. Though an 'approach' event is reasonable to infer as well, it is not necessary for a complete description of the scenario, so its absence is not mentioned. VOX also makes inferences as to who the missing actors were. Other semantic and syntactic errors are reported similarly. The way this text was analyzed is a direct result of the way in which the scenario was entered by using MACRO FRAME.

7.0 IMPLEMENTATION AND LIMITATIONS OF CONCEPTUAL GRAMMAR

During language analysis, a parse tree whose nodes are Conceptual Grammar concepts is constructed. To form the meaning representation, a case-based scheme augments the parse tree. Meaning representation schemes such as conceptual dependency or predicate logic are not claimed to be unnecessary and could be implemented on the basis of the information provided by the Conceptual Grammar representation. Neither Conceptual Grammar nor any other single representation scheme can provide a 'complete understanding' of an utterance, but it is claimed that Conceptual Grammar can be a central component of a multifaceted knowledge representation.

Conceptual Grammar deals exclusively with conceptual and verbally oriented knowledge. It does not serve to represent models of objects, mechanisms, or states of the world, although it can be used to represent the verbalizable or conceptual aspect of such models. Conceptual Grammar assumes that the meaning of concepts must ultimately come from the models they are associated with. As an example, Conceptual Grammar may represent the concept BOOK and a variety of definitional and conceptual information about this concept. But part of the meaning of the concept BOOK must ultimately derive from a real-world model of a book (tactile, visual, etc.) and of how a book interacts with other objects in the physical world. Mental models, belief systems, and models of human beings are just a few types of knowledge that cannot be completely defined within Conceptual Grammar.

Conceptual Grammar provides insights into the notions of knowledge hierarchies and degrees of abstraction. The current development has served to show potential, rather than actual, coverage of linguistic and world knowledge. A very small part of English syntax is handled by Conceptual Grammar. No effort has yet been made to completely cover the major constructions, including those involving modality, negation and relativization, to name a few. In addition, knowledge about selectional restrictions is not currently expressed in Conceptual Grammar rules. Sager (81) has developed a fairly systematic treatment of major classes of such restrictions; we intend to systematize them within the Conceptual Grammar framework as well. Among the more general types of verbalizable knowledge that we would like to represent, the implemented Conceptual Grammar does not yet make use of attributes, goals, plans, states, and themes.

So far, Conceptual Grammar has made almost exclusive use of functional semantics. That is, concepts are defined largely by the contexts (phrases, events, frames) in which they are used. Definitional knowledge is not yet used to a great extent, although semantic hierarchies are one type of definition. Other kinds of definitional knowledge could be incorporated into Conceptual Grammar.

8.0 OTHER EXTENSIBILITY WORK

Automatic extensibility is a relatively unexplored area of natural language processing. While many systems and knowledge representations may be domain-independent and extensible in principle, this usually has not been demonstrated by implementation.

The UC system of Wilensky (Wilensky, et al. 84), which is based on PHRAN, has a UC Teacher component by which a user can add knowledge by telling it new facts. Unlike VOX, the UC Teacher makes use of simple definitional knowledge provided by the user. Most knowledge must still be added to PHRAN by editing complex Lisp data structures, however. Also, the UC Teacher is passive -- it has no capacity to ask the user for information about a new definition.

The LIFER system (Hendrix 77) , which uses a semantic grammar much like PHRAN patterns, has a component for adding new patterns. As in PHRAN, the role of pure syntax in LIFER is minimal, and new semantic information must be added programmatically.

KLAUS (Haas 80, Grosz and Stickel 83), like the UC Teacher, functions by the user telling the sytem new facts. KLAUS also takes an active role to ensure that the new knowledge is being acquired completely and in linguistically correct fashion. Currently, KLAUS can interact with a more linguistically naive user than VOX.

Both KLAUS and UC attempt to interact in English with the user, though both interfaces are fairly crude, and require much knowledge about what language forms the system can handle, in addition to knowledge about concepts the system understands. VOX currently interacts in a canned manner, displaying menus and short-answer questions so as to minimize typing by the user. Both kinds of interface are important in maximizing the user's convenience.

Some specialized systems use automatic extensibility tools as well. For example, TEAM (Grosz 83) provides natural language interfaces for database systems. The acquisition component of TEAM interactively gains knowledge about how users phrase their queries in natural language.

In the dialogues to acquire verbs, nouns, and other parts of speech, VOX, TEAM, and KLAUS are fairly similar in principle. Yet VOX alone has the capacity to acquire and make use of knowledge about scenarios in a systematic way.

9.0 APPENDIX: PROGRAM STATISTICS

VOX is written in UCI MLISP, an Algol-like dialect of Lisp. The code occupies more than 12,000 lines of UCI MLISP. The extensibility system uses 6,000 lines. The rest is split among the analyzer, database manager, and other utilities.

The knowledge base of VOX consists of a database file system and a dictionary file system. The database has about 14,000 lines of Lisp statements. The dictionary has 25,000 entries, only a small percentage of which have database information. The autoextensibility system has 14,000 lines of commands for knowledge addition.

There are over 300 vocabulary-related database entries, each of which holds all the conjugations, parts of speech, and senses of a word. There are about 100 phrases of length > 1 (see the section 'Concepts and Phrases') in the database. The number of grammar rules is about 3,000.

The analyzer uses about one-half minute of CPU time per word analyzed. This includes analysis, error detection, and text rewording. Little has been done to optimize the analyzer.

Work on VOX has been carried out on a small DEC20 computer at the University of California, Irvine, by Amnon Meyers. The system has also been run at the Naval Ocean Systems Center on a larger DEC20; a translator to convert the MLISP code to Zetalisp is currently being developed so that the system can run on Symbolics Lisp machines. It is expected that moving the system to the Lisp machines will result in improved response time and will permit integration of the analyzer and extender, which will increase the practicality of the system. The linguistic capabilities of the system will be greatly enhanced as projects are undertaken to combine the parsing and meaning construction algorithms in the analyzer and to increase the syntactic and semantic coverage of the system.

REFERENCES

Granger, R.H., et al. 1983. NOMAD: A Naval Message Understanding System. Technical Report 209. Artificial Intelligence Project. U Cal Irvine.

Granger, R.H. 1983. The NOMAD System: Expectation-Based Detection and Correction of Errors During Understanding of Syntactically and Semantically Ill-Formed Text. Am J Comp Linguist 9(3-4):188-196.

Grosz, B.J., and M.E. Stickel. 1983. Research on Interactive Acquisition and Use of Knowledge. Final Report. SRI Proj 1894. Artificial Intelligence Center. SRI International.

Grosz, B.J. 1983. TEAM: A Transportable Natural Language Interface System. In Conference on Applied Natural Language Processing. 39-45. Assn for Comp Linguist. Santa Monica.

Haas, N., and G.G. Hendrix. 1980. An Approach to Acquiring and Applying Knowledge. In Nat Conf on Artificial Intelligence. AAAI-80. 235-239. Stanford, CA.

Hendrix, G.G. 1977. The LIFER Manual. Tech Rpt 138. Artificial Intelligence Center. SRI International.

Meyers, A. 1983. Conceptual Grammar. Tech Rpt 215. Artificial Intelligence Project. U Cal Irvine.

---------. 1985a. VOX: An Extensible Natural Language Processor. Tech Rpt 85-02. Artificial Intelligence Project. U Cal Irvine.

---------. 1985b. The VOX Conceptual Grammar. Internal document. Artificial Intelligence Project. U Cal Irvine.

Riesbeck, C.K. 1975. Conceptual Analysis. In Conceptual Information Processing, ed R.C. Schank, 83-156. Elsevier North-Holland, Inc.

Schank, R.C., and R.P. Abelson. 1977. Scripts, Plans, Goals, and Understanding. Lawrence Erlbaum, Hillsdale, N.J.

Wilensky, R., and Y. Arens. 1980. PHRAN: A Knowledge-Based Approach to Natural Language Analysis. Memo UCB/ERL M80/34, Electronics Research Laboratory, Coll of Engr, U Cal Berkeley.

Wilensky, R., et al. 1984. Talking to UNIX in English: An Overview of UC. CACM 27(6):574-593.

END

DTIC

5-86